CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Mailing Label Number EL655301165US

Date of Deposit: March 2, 2001

I hereby certify that this correspondence is being deposited with the United States Postal "Express Mail Post Office to Addressee" service under 37 CFR 1.10(c) on the date indicated above and is addressed to:

BOX PCT
Assistant Commissioner for Patents
Washington DC 20231

Case Number: **P01,0041**
Applicant(s): **Erwin Thurner**

International Application No.	**PCT/DE99/02753**
International Filing Date	**01 SEPTEMBER 1999**
Priority Date Claimed	**02 SEPTEMBER 1998**

Title: **Method for Determining a Graphic Structure of a Technical System and Arrangement and Set of Arrangements for Determining a Graphic Structure**

Enclosed are the following documents:
International application as filed, drawings attached;
English Translation, translated drawings attached;
Annexes;

PTO 1390 in duplicate;
Executed Declaration;

Amendment "A" prior to action and Appendix "A";
Information Disclosure Statement, PTO 1449, Search Report, 04 References;
Submission of Drawings, 5 sheets of drawings, Figures 1-5;

Substitute Specification and Substitute Specification mark-up;

Fee: $860.00;
Postcard.

Craig Thurner

Signature of person mailing documents and fees

09/786388 PCT/DE 99/02753

BUNDESREPUBLIK DEUTSCHLAND

REC'D 26 NOV 1999
WIPO PCT

EU

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

DE 99/2753

Bescheinigung

Die Siemens Aktiengesellschaft in München/Deutschland hat eine Patentanmeldung unter der Bezeichnung

„Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems und Anordnung sowie Satz von Anordnungen zur Bestimmung einer Graphen-Struktur"

am 2. September 1998 beim Deutschen Patent- und Markenamt eingereicht.

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

Die Anmeldung hat im Deutschen Patent- und Markenamt vorläufig das Symbol G 06 F 17/50 der Internationalen Patentklassifikation erhalten.

München, den 20. Oktober 1999
Der Präsident des Deutschen Patent- und Markenamts
Im Auftrag

Nietiedt

Aktenzeichen: 198 39 972.3

A 9161
06.90
11/98

Beschreibung

Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems und Anordnung sowie Satz von Anordnungen zur Bestimmung einer Graphen-Struktur

Es ist bekannt, verschiedene technische Systeme mittels einer graphischen Struktur zu beschreiben.

Aus [1] ist für ein solches technisches System, einer elektronischen Schaltung, bekannt, die elektrische Schaltung in Form einer graphischen Struktur mit Elementen, die eine elektronische Schaltung beschreiben, zu bestimmen.

Elemente einer Graphen-Struktur im Rahmen einer Schaltungssimulation sind Symbole, die elektronische Bauelemente symbolisieren, beispielsweise einen Widerstand, einen Kondensator, eine Induktivität, einen Transistor, einen Operationsverstärker oder andere, aus diesen Elementen zusammengesetzte elektronische Bauelemente.

Bei dem aus [1] bekannten Verfahren und der aus [1] bekannten Anordnung werden von einem Editor-Programm einem Benutzer zur Verfügung gestellte Elemente zur graphischen Beschreibung einer elektronischen Schaltung ausgewählt derart, daß mit den ausgewählten Elementen das technische System "elektronische Schaltung" beschrieben wird. Die Elemente werden von dem Editor-Programm dargestellt.

Eine Graphen-Struktur beschreibt einen Graph G (= V, E, Ψ), der eine endliche, nicht leere Menge V ($v \in V$ bezeichnen Knoten des Graphen G) aufweist sowie eine endliche Menge E ($e \in E$ bezeichnen Kanten des Graphen G). Die Knoten und Kanten des Graphen werden verknüpft durch eine Inzidenzfunktion Ψ, die gemäß folgender Vorschrift gebildet wird:

$$\Psi: E \rightarrow \{\{i, j\} | i, j \in V\} \qquad (1)$$

(i)	$S = \{ s1 , s2 , \ldots , sn \}$	Menge von Stellen
(ii)	$T = \{ t1 , t2 , \ldots , tm \}$	Menge von Transitionen
(iii)	$S \cap T = \varnothing$	S und T disjunkt (die Knotenmenge besteht aus S und T)
(iv)	$F \subseteq (S \times T) \cup (T \times S)$	Flußrelation

Nachteilig an den bekannten Verfahren und Anordnungen ist insbesondere, daß jeweils anwendungsabhängig nur für eine spezielle Anwendung vorgesehene Elemente eines Graphen zur Bestimmung der graphischen Struktur eines technischen Systems zur Verfügung gestellt werden. So kann mit dem Editor-Programm aus [1] lediglich eine Auswahl unter Elementen zur Beschreibung einer elektronischen Schaltung und bei dem Editor-Programm aus [2] lediglich eine Auswahl aus Elementen zur Beschreibung eines Petri-Netzes erfolgen.

Ein solches bekanntes Editor-Programm ist somit äußerst unflexibel für den Fall, daß ein Anwender unterschiedliche Arten einer graphischen Struktur zur Beschreibung eines technischen Systems einsetzen möchte. Für jede spezielle Anwendung muß dann ein eigenes, für die Anwendung angepaßtes Editor-Programm entwickelt werden, was zu erheblichen Entwicklungskosten führt.

Somit liegt der Erfindung das Problem zugrunde, ein Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems sowie eine Anordnung und einen Satz mehrerer Anordnungen zur Bestimmung einer Graphen-Struktur anzugeben, welche gegenüber den bekannten Verfahren und Anordnungen eine verbesserte Flexibilität aufweist.

Das Problem wird durch das Verfahren, die Anordnung sowie den Satz von Anordnungen gemäß den Merkmalen der unabhängigen Patentansprüche gelöst.

a) eine erste Anordnung, die einen Speicher aufweist, in dem eine Menge mehrerer unterschiedlicher Graphen-Struktur-Dateien gespeichert sind, wobei in einer Graphen-Struktur-Datei jeweils angegeben ist, welche Elemente zu deren Darstellung ausgewählt werden können, um einen Graphen zu bilden, und

b) eine mit der ersten Anordnung gekoppelte zweite Anordnung, die folgende Komponenten aufweist:
   - eine Auswahleinheit, mit der eine Graphen-Struktur-Datei aus der Menge der Graphen-Struktur-Dateien ausgewählt werden kann,
   - ein Editor-Programm, mit dem unter Verwendung aus der Menge der Graphen-Struktur-Dateien ausgewählten Graphen-Struktur-Datei ein Graph mit Elemenenten der ausgewählten Graphen-Struktur-Datei bestimmt werden kann, womit die Graphen-Struktur bestimmt ist,
   - eine mit dem Editor-Programm gekoppelte Darstellungskomponente, mit der die bestimmte Graphen-Struktur dargestellt werden kann.

Durch die Erfindung wird ein gegenüber den bekannten Verfahren und Anordnungen sehr flexibles Verfahren und eine sehr flexible Anordnung zur Bestimmung einer graphischen Struktur angegeben, welche schnell und unkompliziert an neue Anwendungsszenarien bzw. an bestehende Anwendungsszenarien besser angepaßt werden kann.

Auf diese Weise werden verschiedene Arten von Strukturen, die als Graph darstellbar sind, mit einem Verfahren bzw. mit einer Anordnung auf flexible, kostengünstige und einfache Weise bearbeitbar.

Bevorzugte Weiterbildungen der Erfindung ergeben sich aus den abhängigen Ansprüchen.

Das technische System ist vorzugsweise eine elektronische Schaltung oder eine technische Anlage.

Figur 4 ein Ablaufdiagramm, in dem die Verfahrensschritte des Verfahrens gemäß einem Ausführungsbeispiel dargestellt sind;

Figur 5 ein Satz mehrerer Anordnungen, die gemäß einem zweiten Ausführungsbeispiel über ein Kommunikationsnetz miteinander gekoppelt sind.

Fig.1 zeigt eine Anordnung 100 mit einer Menge 101 mehrerer unterschiedlicher Graphik-Struktur-Dateien 102, 103, 104, 105. Jede Graphik-Struktur-Datei 102, 103, 104, 105 ist als dynamisch bindbare Datei (dynamic link library) ausgestaltet.

Von einem Benutzer 106 wird über eine mit einem Editor-Programm 107 verbundenen Auswahlkomponente 108 (Tastatur und/oder Computermaus) eine Graphik-Struktur-Datei 102, 103, 104, 105 ausgewählt.

Die ausgewählte Graphik-Struktur-Datei, in diesem Ausführungsbeispiel eine erste Graphik-Struktur-Datei 103, wird dynamisch in das Editor-Programm 107 eingebunden.

Nach Einbindung in das Editor-Programm 107 ist über eine mit dem Editor-Programm 107 verbundene Darstellungskomponente 109 dem Benutzer 106 auf einem Bildschirm 110 eine Menge 111 von auswählbaren Elementen 112, 113, 114, die in der ersten Graphik-Struktur-Datei 103 angegeben sind, als auswählbare Elemente zur Bestimmung eines weiteren beschriebenen Graphen dargestellt. Ferner sind in diesem Ausführungsbeispiel gemäß der ersten Graphik-Struktur-Datei 103 ein erstes Überprüfungsprogramm 115 sowie ein zweites Überprüfungsprogramm 116 in dem Editor-Programm 107 eingebunden und werden dem Benutzer 106 zur Auswahl zur Verfügung gestellt.

Jede Graphik-Struktur-Datei 102, 103, 104, 105 weist jeweils eine Menge auswählbarer Elemente für die jeweilige Art von Graphen auf, wobei jeweils eine Graphik-Struktur-Datei vorge-

Über eine Menge 206 weiterer, im weiteren näher erläuterter Auswahlelemente ist die Auswahl und Bearbeitung spezifischer Elemente für ein Petri-Netz 201 dem Benutzer 106 zur Verfügung gestellt.

Ein zweites Auswahlelement 207 ist beschrieben durch ein leeres Rechteck und symbolisiert eine zeitbehaftete Transititon.

Ein drittes Auswahlelement 208 symbolisiert eine zeitlose Transition, die als ausgewählte Transisitions-Elemente 220, 221 und 222 in dem Petri-Netz 201 dargestellt sind.

Ein viertes Auswahlelement 209 symbolisiert eine Kante, die in diesem Ausführungsbeispiel eine gerichtete Kante ist.

Ein fünftes Auswahlelement 210 symbolisiert eine gemäß der Strukturregeln eines Petri-Netzes 201 bezeichnete verbotene Kante.

Ein sechstes Auswahlelement 211 symbolisiert eine Stelle, wobei jeweils ein Stellen-Element 223, 224, 225, 226 in dem Petri-Netz 201 dargestellt sind. Die Stellen-Elemente 223, 224, 225 und 226 sind mit den Transition-Elementen 220, 221, 222 über Kanten 227, 228, 229, 230, 231 und 232 verbunden.

Ein siebtes Auswahlelement 212 symbolisiert die Möglichkeit, eine Kombination mehrerer Elemente des Petri-Netzes zu einem Gesamtelement zusammenzufassen.

Ein achtes Auswahlelement 213 symbolisiert einen Eingang des Petri-Netzes 201 und ein neuntes Auswahlelement 214 symbolisiert einen Ausgang eines Petri-Netzes 201.

Den Kanten sowie den einzelnen Knoten, das heißt den Elementen des Petri-Netzes 201 sind textuelle Informationen 251, 252, 253, 254, 255, 256, 257, 258, 259, 260 und 261 zugeord-

auf, die jeweils über Kanten 317 miteinander verbunden sind. Ferner ist ein Masseanschluß 318 in Fig.3 dargestellt. Den einzelnen Schaltungselementen ist textuelle Information 319, 320, 321, 322, 323, 324, 325, 326 zugeordnet zur näheren Erläuterung der elektronischen Schaltung 310.

Fig.4 zeigt zur Verdeutlichung des Verfahrens das Verfahren in seinen Verfahrensschritten.

In einem ersten Schritt (Schritt 401) wird eine Graphik-Struktur-Datei 102, 103, 104, 105 aus einer Menge 101 von Graphik-Struktur-Dateien 102, 103, 104, 105 ausgewählt.

In einem zweiten Schritt (Schritt 402) erfolgt eine Auswahl von Elementen, die gemäß der Graphik-Struktur-Datei 102, 103, 104, 105, die in dem Schritt zuvor (Schritt 401) ausgewählt wurde, zur Verfügung stehen.

Die ausgewählten Elemente werden von dem Editor-Programm 107 in einem weiteren Schritt (Schritt 403) dargestellt.

Fig.5 zeigt einen ersten Rechner 500 mit einem Speicher 502 und einem Prozessor 503, die jeweils über einen Bus 504 miteinander und mit einer Eingangs-/Ausgangsschnittstelle 501 verbunden sind.

Über die Eingangs-/Ausgangsschnittstelle 501 ist der erste Rechner 500 mit einem Bildschirm 505, einer Tastatur 506 sowie einer Computermaus 507 verbunden.

Ferner ist der erste Rechner 500 über ein Kommunikationsnetz 560, in dem Ausführungsbeispiel ein ISDN-Netz (Integrated Services Digital Network) mit weiteren Rechnern 510, 520, 530, 540 und 550 verbunden.

In dem ersten Rechner 500 ist die Menge 101 der Graphik-Struktur-Dateien 102, 103, 104, 105 gespeichert.

diglich von der jeweiligen Art des zu bestimmenden Graphen abhängig.

Das technische System kann beispielsweise auch eine technische Anlage sein, die durch den Graphen in ihrem Verhalten oder in ihrer Struktur beschreibbar ist.

Das Editor-Programm sowie der mit dem Editor-Programm dargestellte Graph kann im Rahmen einer Simulation des technischen Systems eingesetzt werden.

Im weiteren ist eine Realisierung des oben beschriebenen Ausführungsbeispiels angegeben, geschrieben in der Programmiersprache C, wobei die Realisierung in drei Dateien aufgegliedert ist:

1. Initialisierungs-Datei:

```
package interfaces;

import java.io.*;
import java.util.*;
import java.awt.*;

import etc.*;
import elements.*;
import mmi.*;
import tools.*;

public class Initialisierung {
  GraphEditor editor;
  // Der hat die Tokens aus der Datei
  StreamTokenizer token;
  // Hier kommen alle erlaubten Knoten und Kanten aus der
  // .lgc Datei rein.
  // Die Einträge werden mit den Namen der Objekte referenziert
  Hashtable gobjekte;
  // Die aktuelle .lgc Datei
  //String configFile;
  // steht jetzt bei den Einstellungen
  /**
   * Hier stehen alle Attribute drin.
   */
  Hashtable attributNamen;
  /**
   * hier kommen die Einträge für das Menue Tools
   * hinein.
   */
  Hashtable tools;

  public Initialisierung(GraphEditor editor) {
    this.editor = editor;
    gobjekte = new Hashtable();
    attributNamen = new Hashtable();
    tools = new Hashtable();
  }

  /**
   * Diese Methode würde die erste Initialisierungsdatei
   * einlesen für die Einstellugen der Farben, Schriften...
   * Aber ich darf leider nicht.
   */
  /*
   public void readFirst(String name) {
    String configFile = new String(name);
    int c;
    //Properties properties = new Properties();
    //properties = System.getProperties();
    //filename = new String(".." + properties.getProperty("file.separator") + configfile);
    try {
      File file = new File(configFile);
      //FileInputStream in = new FileInputStream(file);
      FileReader in = new FileReader(file);
      token = new StreamTokenizer(in);

      //Einstellen der Optionen für token
      token.eolIsSignificant(true);
      token.quoteChar('"');
      //token.quoteChar('\'');
      //token.quoteChar('{');
      token.quoteChar('}');

      //Überlese { und , und ;
      token.whitespaceChars('{','{');
      token.whitespaceChars(',',',');
      token.whitespaceChars(';',';');

      boolean fertig = false;
      while (!fertig) {
        switch (c=token.nextToken()){
```

```
                if
(token.sval.equals("SHORTCUTS"))
{
                  while (c != '}')
{
c=token.nextToken();
                    if (c == '"')
{
                      String
mpunkt = token.sval;
//System.out.print("MENUPUNKT " +
token.sval);
c=token.nextToken();
                      String icon1
= token.sval;
//System.out.print("ICON1 " + to-
ken.sval);
c=token.nextToken();
                      String icon2
= token.sval;
//System.out.println("ICON2 " +
token.sval);
                      edi-
tor.getShortcutleiste().addShortB
utton();
                    }
                  }
                   break;
                }
               if
(token.sval.equals("ACCELERATOR")
) {
                 while (c != '}')
{
c=token.nextToken();
                   if (c == '"') {
                     String la-
bel = token.sval;
//System.out.print("MENUPUNKT " +
token.sval);
c=token.nextToken();
                     if (c ==
StreamTokenizer.TT_WORD) {
                     char cut =
token.sval.charAt(0);
//System.out.println(" TASTEN " +
cut);
                     edi-
tor.getMenueleiste().addShortcutT
oVector(label, cut);
                     }
                  }
                  }
                  break;
                }
                if
(token.sval.equals("WINDOWSIZE"))
{
c=token.nextToken();
                    int x
=(int)token.nval;
c=token.nextToken();
c=token.nextToken();
                    int y
=(int)token.nval;
//size.setSize(x,y);
                    break;
                }
                if
(token.sval.equals("WINDOWPOSITIO
N")) {
c=token.nextToken();
                   int x
=(int)token.nval;
c=token.nextToken();
c=token.nextToken();
                    int y
=(int)token.nval;
//location.setSize(x,y);
                    break;
                }
                 if
(token.sval.equals("AUTHOR")) {
c=token.nextToken();
                 if (c == '"') {
                     Sy-
stem.out.println("AUTHOR " + to-
ken.sval);
                   }
                   break;
                 }
                if
(token.sval.equals("TOOLS")) {
                 while (c != '}')
{
c=token.nextToken();
                    if (c == '"')
{
                      String pfad
=new String(token.sval);
//System.out.println("TOOL " +
token.sval);
```

```
                }
                if
(token.sval.equals("ACCELERATOR")
) {
                  Sy-
stem.out.println("Lese Accelera-
tor");
                  readAccel();
                  break;
                }
                default:
              }
            }

    in.close();
    System.out.flush();
    System.out.println("EINLESEN
DER DATEI " +configFile + "
FERTIG!");
    //und wichtig für die Anzei-
ge:
    setLayer();
    setAttributNames();
    } catch
(FileNotFoundException e) {
        System.err.println( con-
figFile + " is not found");
    } catch (IOException e) {
       e.printStackTrace();
    }
  }

  private void readToolbar(String
lgcPath) {
    int c ='{';
    gobjekte.clear();
    //System.out.println("Jetzt
kommt die Toolbar");
    try {
      while (c != '}') {
        switch
(c=token.nextToken()){
          case StreamTokeni-
zer.TT_WORD:
            if
(token.sval.equals("NODE")) {

//System.out.println("Lese Kno-
ten");
              readNode(lgcPath);
              break;
            }
            if
(token.sval.equals("EDGE")) {

//System.out.println("Lese Kan-
te");
              readEdge(lgcPath);
              break;
            }
          default:
        }
        //c=token.nextToken();
        //System.out.println("IN
der TOOLBAR " +c );
      }
    } catch (IOException e) {
      e.printStackTrace();
    }
    //System.out.println("Fertig
Toolbar");
  }


  private void readNode(String
lgcPath) {
    int c ='{';
    String typname = new
String();
    String image = new String();
    Vector ecken = new Vector();
    Vector konnektoren = new Vec-
tor();
    Vector konnektorNamen = new
Vector();
    Attribute attribute = new
StandardAttribute();
    Color color = new Co-
lor(255,255,255);
    //System.out.println("Ein
Knoten");
    try {
      while (c != '}') {
        switch (c){
          case StreamTokeni-
zer.TT_WORD:
            // Wird nicht mehr be-
nötigt
            // if
(token.sval.equals("TYPE")) {
            //
c=token.nextToken();
            //   Sy-
stem.out.println("Lese TYPE" +
token.sval);
            //   break;
            // }
             if
(token.sval.equals("NAME")) {
c=token.nextToken();
              typname = new
String(token.sval);
              // Sy-
stem.out.println("Lese NAME" +
typname);
              break;
             }
             if
(token.sval.equals("ATTRIBUTES"))
{
              attribute = new
StandardAttribute();
```

```
attribute);
                kno-
ten.setColor(color);
                // Sy-
stem.out.println("Setze Farbe " +
color);
                // Erzeuge Button
mit Werzeug für Werkzeugleiste
                // Der Button greift
über den typnamen auf den richti-
gen
                // Knoten zu.
                ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typname,

new KnotenTool(editor,typname),

editor.getToolbar());
                edi-
tor.getToolbar().addToolButton(b)
;
                // Eintrag in die
Hashtabelle
                gobjek-
te.put(typname,knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

                 break;
               }
               if
(token.sval.equals("FILLEDOVAL"))
{
                 int breite=10;
                 int hoehe=10;
                 while
((c=token.nextToken()) != '}') {
                   breite =
(int)token.nval;

c=token.nextToken();
                   hoehe =
(int)token.nval;
                   // Sy-
stem.out.println("Lese OVAL_FILL"
+ token.nval);
                 }
                 // jetzt sollten
alle Daten da sein, und es
                 // kann ein Knoten-
prototyp erzeugt werden.
                 GraphObjekt knoten
= new FilledOvalKnoten(typname,

hoehe,

breite,
```

```
konnektoren,

konnektorNamen,

attribute);
                kno-
ten.setColor(color);
                // Sy-
stem.out.println("Setze Farbe " +
color);
                // Erzeuge Button
mit Werzeug für Werkzeugleiste
                ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typname,

new KnotenTool(editor,typname),

editor.getToolbar());
                edi-
tor.getToolbar().addToolButton(b)
;
                // Eintrag in die
Hashtabelle
                gobjek-
te.put(typname,knoten);

//System.out.println("In Hashta-
belle: " + gobjekte);

                 break;
               }
               if
(token.sval.equals("OVAL")) {
                 int breite=10;
                 int hoehe=10;
                 while
((c=token.nextToken()) != '}') {
                   breite =
(int)token.nval;

c=token.nextToken();
                   hoehe =
(int)token.nval;
                   // Sy-
stem.out.println("Lese OVAL" +
token.nval);
                 }
                 // jetzt sollten
alle Daten da sein, und es
                 // kann ein Knoten-
prototyp erzeugt werden.
                 GraphObjekt knoten
= new OvalKnoten( typname,

hoehe,

breite,
```

```
    Color color = new Co-
lor(255,255,255);
    try {
      while (c != '}') {
        switch (c){
          case StreamTokeni-
zer.TT_WORD:

            if
(token.sval.equals("NAME")) {

c=token.nextToken();
              typname = new
String(token.sval);
              // Sy-
stem.out.println("Lese NAME" +
typname);
              break;
            }
            if
(token.sval.equals("ATTRIBUTES"))
{
              attribute = new
StandardAttribute();
              while
((c=token.nextToken()) != '}') {
                String aname =
new String(token.sval);
                c = to-
ken.nextToken();
                String wert = new
String(token.sval);
                attribu-
te.addAttribut(aname,wert,true);
                attributNa-
men.put(aname,aname);
                // Sy-
stem.out.println("Lese Attribut-
te" + attribute);
              }

              break;
            }
            if
(token.sval.equals("IMAGE")) {

c=token.nextToken();
              image = new
String(token.sval);
              // Sy-
stem.out.println("Lese IMAGE" +
image);
              break;
            }
            if
(token.sval.equals("ARROW")) {
              int radius = 10;
              int winkel = 10;
              while
((c=token.nextToken()) != '}') {
                radius =
(int)token.nval;

c=token.nextToken();
                winkel =
(int)token.nval;
                // Sy-
stem.out.println("Lese Arrow" +
radius+ winkel);
              }
              // jetzt sollten
alle Daten da sein, und es
              // kann ein Kanten-
prototyp erzeugt werden.
              GraphObjekt kante =
new PfeilKante(typname,

radius,

winkel,

attribute);
              kan-
te.setColor(color);
              // Sy-
stem.out.println("Setze Farbe " +
color);
              // Erzeuge Button
mit Werzeug für Werkzeugleiste
              ToolButton b = new
ToolButton(lgcPath + "images/" +
image,

typname,

new KantenTool(editor,typname),

editor.getToolbar());
              edi-
tor.getToolbar().addToolButton(b)
;
              // Eintrag in die
Hashtabelle
              gobjek-
te.put(typname,kante);

//System.out.println("In Hashta-
belle: " + gobjekte);

              break;
            }
            if
(token.sval.equals("POINT")) {
              int durch = 10;
              while
((c=token.nextToken()) != '}') {
                durch =
(int)token.nval;
              // Sy-
stem.out.println("Lese Point" +
durch);
              }
              // jetzt sollten
alle Daten da sein, und es
```

```
                break;
              }
            default:
          }//switch
          c=token.nextToken();
          // System.out.println("NAECHSTES TOKEN" + token.sval);
        } //while
        //c=token.nextToken();
      } catch (IOException e) {
         e.printStackTrace();
      }
      // System.out.println("Bende readEdge");

    }//readEdge


    private void readMenu() {
      tools.clear();
      int c = '(';
      try {
        while ((c=token.nextToken()) != ')') {
          //c=token.nextToken();
          String namen = token.sval;
          System.out.println("Jetzt kommt das Menu"+ namen);
          c = token.nextToken();
          String aufruf = token.sval;
          System.out.println("Jetzt kommt das Menu"+ aufruf);
          tools.put(new String(namen), new String(aufruf));
        }
      } catch (IOException e) {
         e.printStackTrace();
      }

    }

    private void readAnalyse() {
      System.out.println("Jetzt kommt die Analyse");
    }

    private void readShorts() {
      System.out.println("Jetzt kommt die Shortcut");
    }

private void readAccel() {
    System.out.println("Jetzt kommen die Accelerators");
    }
```

```
 // private void uebergebe (String mpunkt,String icon1,String icon2) {
 //   public void addBut-
ton(String menuePunkt, String image1, String image2)

  private void uebergebe(String auswahl,String name,String style, int size) {
    int styleInt = 0;
    switch (style.charAt(0)){
      case 'B':
        styleInt = Font.BOLD;
        break;
      case 'P':
        styleInt = Font.PLAIN;
        break;
      case 'I':
        styleInt = Font.ITALIC;
        break;
      default:
        styleInt = Font.PLAIN;
    }
    Font font = new Font(name, styleInt, size);
    switch (auswahl.charAt(0)){
      case 'M':
        edi-
tor.getMenueleiste().setFont(font);
        break;
      case 'P':
        //noch zu Implementiern
        break;
      case 'S':
        edi-
tor.getStatusleiste().setFont(font);
        break;
    }
  }

  private void uebergebe(String auswahl,int r,int g,int b) {
    if (auswahl.equals("PAPER")){
      edi-
tor.getZeichenflaeche().setBackground(new Color(r,g,b));
    }
    if (auswahl.equals("GRID")){
      //noch zu implementiern
    }
    if (auswahl.equals("MENUBGC")){
   // edi-
tor.getMenueleiste().setBackground(new Color(r,g,b));
    }
    if (auswahl.equals("MENUFGC")){
```

```
    /**
     * Liefert alle anzeigbaren
AttributNamen zurück.
     */
    public Enumeration getAttri-
butNames() {
      return attributNamen.keys();
    }

    /**
     * Liefert die maximale Anzahl
der Attribute zurück.
     */
    public int countAttributNa-
mes() {
      return attributNamen.size();
    }

    /**
     * Fügt einen Attribut namen
in die
     * Hashtabel ein.
     */
    public void addAttributName (
String name) {
      attributNamen.put(new
String(name), new String(name));
    }

    /**
     *
     */
    public Hashtable getTools() {
      return tools;
    }

 // public String getConfigFile()
{
  //  return configFile;
 // }
}
```

2. Datei "load"

```
package commands;

import etc.*;
import java.util.*;
import java.awt.*;
import java.io.*;
import interfaces.*;

/**
 * Lädt einen Graphen aus einer
.lgf Datei.
 */
public class Load extends Befehl
{
  Vector undo;

  public Load(GraphEditor edi-
tor){
    super(editor);
    undo=new Vector();
    help =
"<filename.lgf/.lgc/.lgt>";
  }


  public void ausfuehren(String[]
param){
    //System.out.println(param);
    int anzahl = param.length;
    switch (anzahl) {
      case 0  : // bei keinem Ar-
gument tun wir nichts.
                break;
      case 1  : // bei einen Ar-
gument wird erst nachgeschaut!
                if
(param[0].endsWith(".lgc") ||
                   pa-
ram[0].endsWith(".lgf") ||
                   pa-
ram[0].endsWith(".lgt") ) {
                  // wir wurden
von der CommandoZeile aufgerufen
                  File file = new
File(param[0]);

//System.out.println("Der Pfad :
" + file.getParent());

//System.out.println("Der Name :
" + file.getName());
                  prue-
fe(file.getParent()+"/",file.getN
ame());
                } else {
                  //nothing
                }
                break;
      default : //zuviel Parame-
ter
                break;
    }//switch
  }

  public void ausfuehren(String
param){
    edi-
tor.getStatusleiste().show("Load.
..");

((Component)editor).setCursor(Cur
```

```
  }// redo

  /**
   * Diese Klasse wird leider
nicht an
   * die Windows bzw Solaris Kom-
ponente
   * weitergereicht.
   */
  class lgFilter implements Fi-
lenameFilter {
    public boolean accept (File
dir, String name) {
      return ( na-
me.endsWith(".lgf") ||
               na-
me.endsWith(".lgc") ||
               na-
me.endsWith(".lgt")  );
    }
  }
  /**
   * Diese Methode überprüft, ob
die richtige
   * Konfigurationsdatei geladen
ist, ansonsten wird
   * versucht die richtige zu la-
den.(->Editor zurücksetzen)
   * Dannach wird die gewünschte
.lgt oder .lgf Datei
   * geladen.
   */
  private void pruefe (String
pfad, String datei) {
    Einstellungen settings= edi-
tor.getEinstellungen();
   if (datei.endsWith(".lgc")) {
     //System.out.println("eine
lgc Datei");
      File f = new File(pfad +
datei);
      if (f.exists()) {
        settings.appName = Ein-
stellungen.format(datei);
        settings.fileName=" ";
        settings.frameName = set-
tings.fileName+ "  "
+settings.appName + "  "
+settings.copyright;
        settings.configFile = new
String(datei);
        settings.lgcPath = new
String(pfad);
        //wir Starten den Editor
neu
        editor.start();
      } else {
        System.err.println("File
not found : "+ settings.lgcPath +
datei);
      }
    } else if
(datei.endsWith(".lgf")) {
      //System.out.println("eine
lgf Datei");
      File f = new File(pfad +
datei);
      if (f.exists()) {
        settings.fileName = da-
tei;
        // wir holen uns noch den
namen des .lgc Files:
        String config = edi-
tor.getDateischnittstelle().getCo
nfig(pfad + datei);
        //System.out.println("Der
neue Name der Lgc datei " + con-
fig);
        f = new
File(settings.lgcPath + config);
        if (f.exists()) {
          // ist diese lgc Datei
schon geladen?
          if
(settings.configFile.equals(confi
g)) {
            //wir muessen nur die
lgf Datei laden
            edi-
tor.getDateischnittstelle().load(
pfad,datei,editor.getGraph());
            settings.frameName =
settings.fileName+ "  "
+settings.appName + "  "
+settings.copyright;
            ((Frame)editor). set-
Title(settings.frameName);
          } else {
            // wir müssen auch
die Konnfigurationsdatei laden
            settings.appName =
Einstellungen.format(config);
            settings.configFile =
new String(config);
            settings.frameName =
settings.fileName+ "  "
+settings.appName + "  "
+settings.copyright;
            //wir Starten den
Editor neu
            editor.start();
            edi-
tor.getDateischnittstelle().load(
pfad,datei,editor.getGraph());
          }
        } else {
          Sy-
stem.err.println("File not found
: " + settings.lgcPath + config);
        }
      } else {
        System.err.println("File
not found : " + pfad + datei);
```

```
 * Die Toolbar ermöglicht das hinzufügen und entfernen
 * von ToolButtons, und deren zugehörigen ActionListener.
 */
public class Toolbar extends Panel {
  GraphEditor editor;
  Tool currentTool;
  ToolButton currentButton;
  int borderSize = 4;
  /**
   * Der Konstruktor erzeugt das AuswahlTool,
   * da dieses immer vorhanden sein sollte.
   */
  public Toolbar(GraphEditor editor){
    this.editor = editor;
    setLayout(new BarLayout(BarLayout.VERTIKAL,2));
    setBackground(editor.getEinstellungen().toolbarBgCo);
    // eine kleine Lücke
    add(new Space(5,24));

    ToolButton b = new ToolButton(editor.getEinstellungen().lgcPath +
"images/auswahl.gif",
"Select",
new AuswahlTool(editor),this);
    setCurrentTool(b.getTool());
    setCurrentButton(b);
    add(b);
    add(new Space(5,24));
  }


  public Insets getInsets() {
    Insets insets = (Insets)(super.getInsets()).clone();
    insets.top += borderSize;
    insets.left += (borderSize+2);
    insets.bottom += borderSize;
    insets.right += (borderSize+2);
    return insets;
  }

  public void paint(Graphics g) {
    super.paint(g);
    Insets insets = super.getInsets();
    int w = getSize().width-insets.left-insets.right;
    int h = getSize().height-insets.top-insets.bottom;

    g.setColor(editor.getEinstellungen().toolbarBgCo);
    for (int i=0; i<borderSize; i++) {
      g.draw3DRect(i+insets.left,i+insets.top,
                   w-2*i-1, h-2*i-1, i<borderSize/2);
    }
  }

  /**
   * Fügt einen ToolButton hinzu.
   */
  public void addToolButton(ToolButton button) {
    add(button);
  }

  /**
   * Entfernt einen ToolButton.
   */
  public void deleteTooleButton(ToolButton button) {
  }

  /**
   * Setzt das aktuelle Tool;
   * wird normalerweise von den ToolButtons aufgerufen.
   */
  public void setCurrentTool(Tool currentTool) {
    this.currentTool = currentTool;
    this.currentTool.reset();
  }

  /**
   * Setzt den aktuellen Button, damit der nächste
   * aktuelle Butten ihn zurücksetzen kann.
   */
  public void setCurrentButton(ToolButton currentButton) {
    if (this.currentButton != null)
      this.currentButton.setUp();
    this.currentButton = currentButton;
    this.currentButton.setDown();

  }

  /**
```

Patentansprüche

1. Verfahren zur Bestimmung einer graphischen Struktur eines technischen Systems,
   a) bei dem aus einer Menge mehrerer unterschiedlicher Graphen-Struktur-Dateien eine Graphen-Struktur-Datei ausgewählt wird, wobei in einer Graphen-Struktur-Datei jeweils angegeben ist, welche Elemente zu deren Darstellung ausgewählt werden können, um das technische System in seiner Struktur graphisch zu beschreiben,
   b) bei dem Elemente ausgewählt werden derart, daß mit den ausgewählten Elementen das technische System beschrieben wird, und
   c) bei dem die Elemente von einem Editor-Programm dargestellt werden, in welches die ausgewählte Graphen-Struktur-Datei eingebunden worden ist, womit die graphische Struktur des technischen Systems bestimmt ist.

2. Verfahren nach Anspruch 1,
bei dem das technische System eine elektronische Schaltung ist.

3. Verfahren nach Anspruch 2,
bei dem das technische System eine technische Anlage ist.

4. Verfahren nach einem der Ansprüche 1 bis 3,
bei dem die Elemente Graphenelemente eines Graphen sind, die das technische System beschreiben.

5. Verfahren nach einem der Ansprüche 1 bis 4,
bei dem die bestimmte graphische Struktur des technischen Systems auf vorgegebene Strukturregeln hin überprüft wird.

6. Anordnung zur Bestimmung einer Graphen-Struktur,
   a) mit einem Speicher, in dem eine Menge mehrerer unterschiedlicher Graphen-Struktur-Dateien gespeichert sind, wobei in einer Graphen-Struktur-Datei jeweils angegeben

- eine Auswahleinheit, mit der eine Graphen-Struktur-Datei aus der Menge der Graphen-Struktur-Dateien ausgewählt werden kann,
- ein Editor-Programm, mit dem unter Verwendung einer aus der Menge der Graphen-Struktur-Dateien ausgewählten Graphen-Struktur-Datei ein Graph mit Elementen der ausgewählten Graphen-Struktur-Datei bestimmt werden kann, womit die Graphen-Struktur bestimmt ist,
- eine mit dem Editor-Programm gekoppelte Darstellungskomonente, mit der die bestimmte Graphen-Struktur dargestellt werden kann.

11. Satz von Anordnungen nach Anspruch 10,
bei dem die erste Anordnung und die zweite Anordnung über ein Kommunikationsnetz miteinander gekoppelt sind.

12. Satz von Anordnungen nach Anspruch 10 oder 11,
bei dem mit dem Graphen eine Struktur eines technischen Systems beschrieben wird.

13. Anordnung nach Anspruch 12,
bei dem das technische System eine elektronische Schaltung ist.

14. Anordnung nach Anspruch 12,
bei dem das technische System eine technische Anlage ist.

115

FIG 1

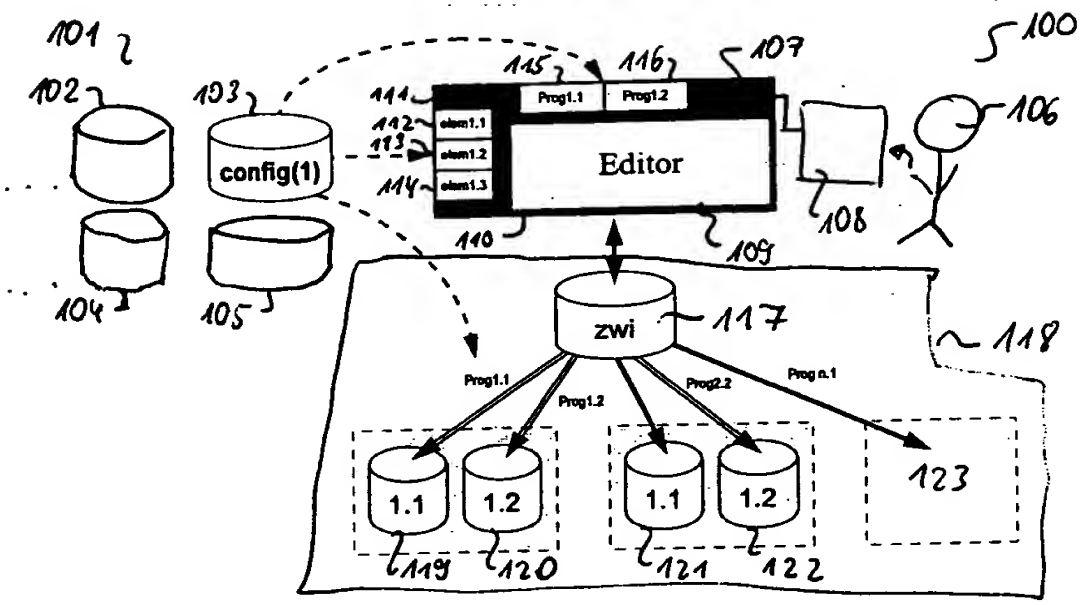

101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
100
config(1)
Editor
Prog1.1
Prog1.2
elem1.1
elem1.2
elem1.3
zwi
Prog1.1
Prog1.2
Prog2.2
Prog n.1
1.1
1.2
1.1
1.2
123
119
120
121
122

FIG 3

315

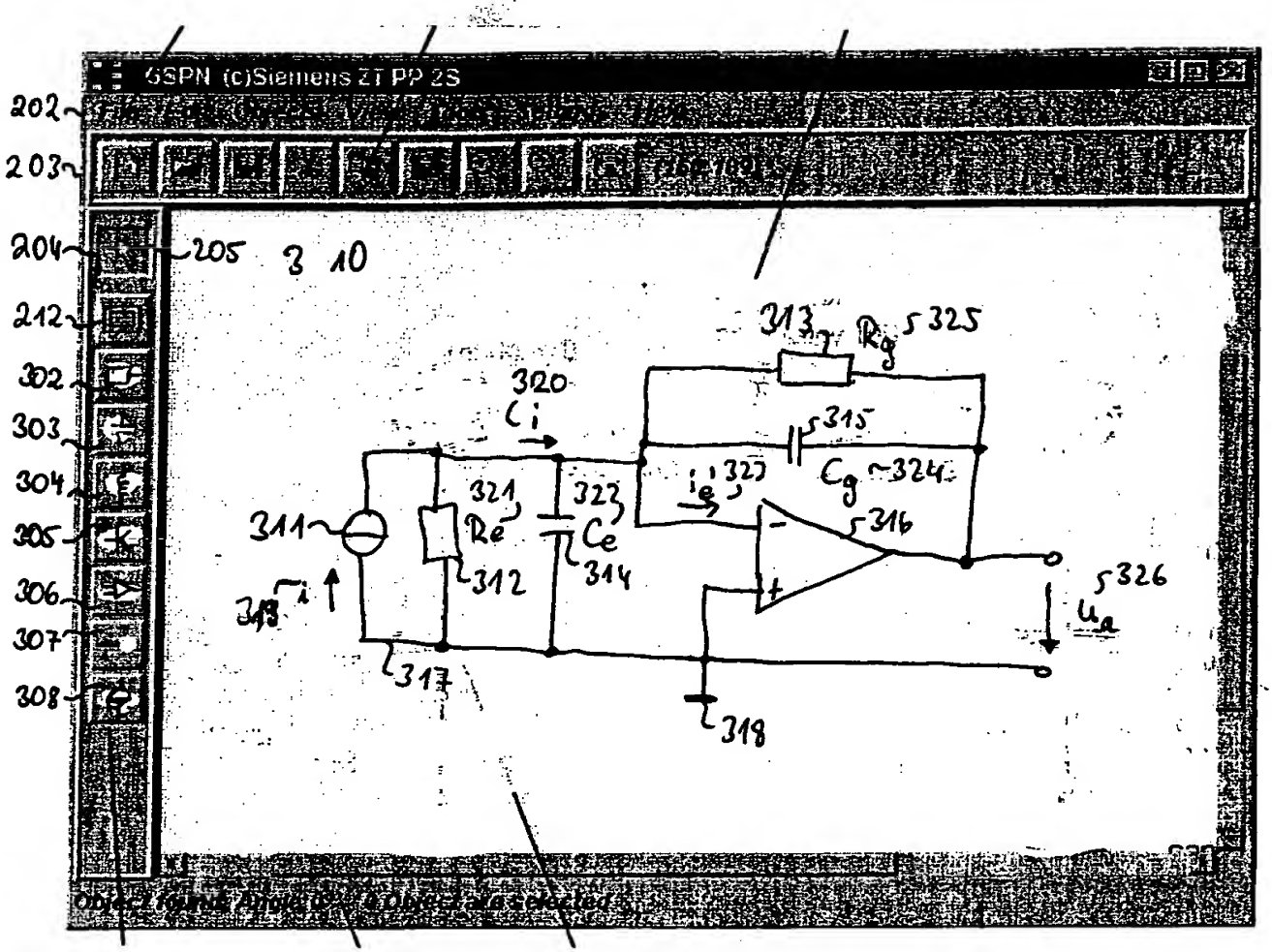
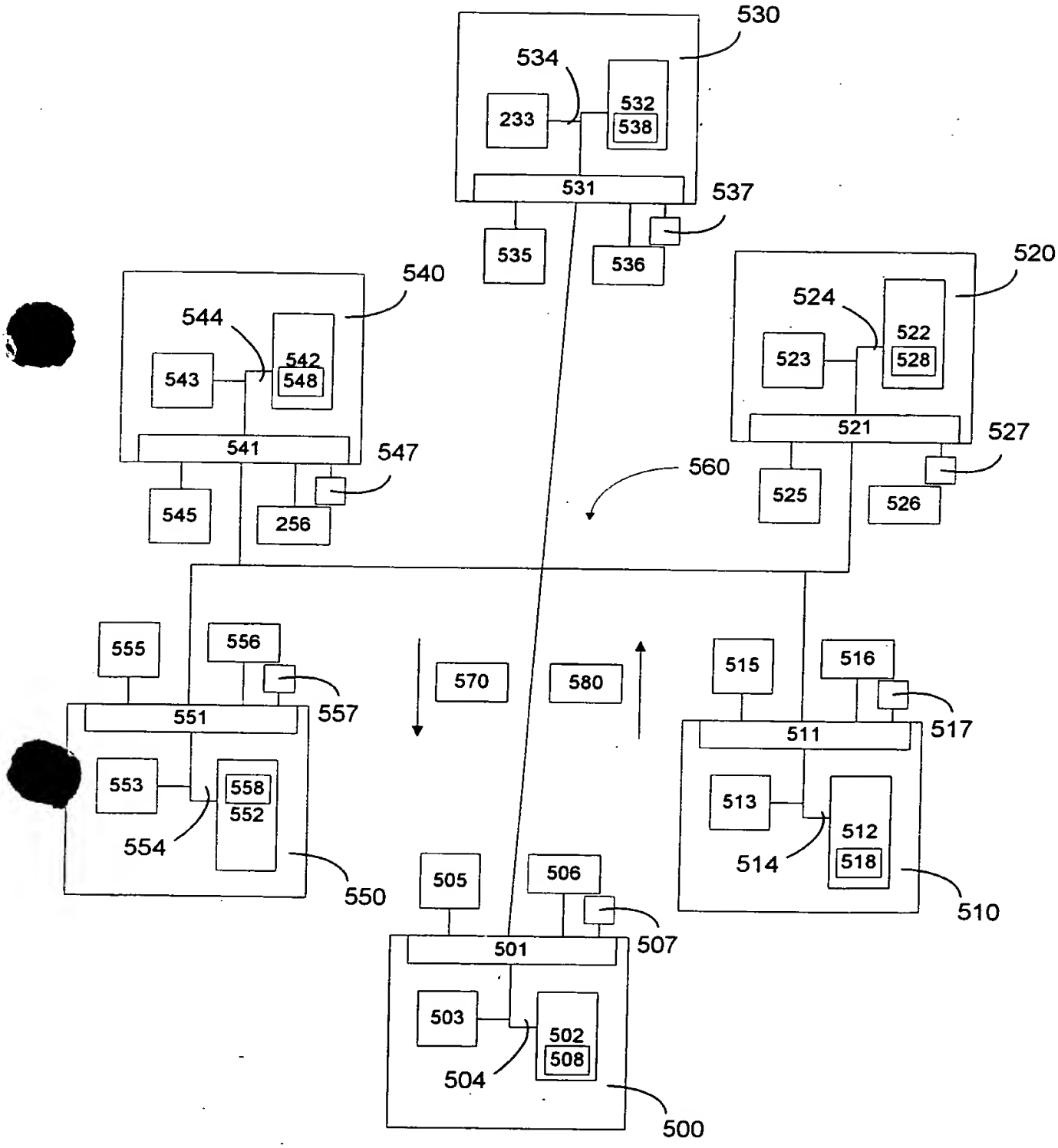

GSPN (c)Siemens ZT PP 2S
202
203
204
205
212
302
303
304
305
306
307
308
301
310
313
Rg
325
320
Ci
315
321
Re
322
Ce
323
ie
Cg
324
316
311
312
314
326
Ua
317
318
Object found. Applic
Object selected

FIG 5


530
534
233
532
538
531
537
535
536
520
524
522
523
528
521
527
525
526
560
540
544
542
543
548
541
547
545
256
555
556
557
570
580
515
516
517
551
511
553
558
552
554
513
512
514
518
550
510
505
506
507
501
503
502
508
504
500